1.2 Complexity hierarchies

Try to answer the following questions:

(1.) Is there a computable function $t: \mathbb{N} \to \mathbb{N}$ s.t. $\underline{\Phi}(t)$ contains all decidable sets?

(2.) Given any computable function $t: \mathbb{N} \to \mathbb{N}$ is there a decidable set A s.t. $A \notin \underline{\Phi}(t)$?

(3) (If (2) is true) Given t, how much greater has t' to be in order to get $\underline{\Phi}(t) \subset \underline{\Phi}(t')$?

1.2.1 Deterministic space

We consider DSPACE = 2-T-DSPACE.

Preliminary remark: 2-T-TM $M = (\Sigma, \Delta, S, \delta, s_+, s_-)$

- Σ is a finite input alphabet
- Δ is a finite internal alphabet
- S is a finite set of states
- $\delta: S \times \Sigma \times \Delta \to S \times \Delta \times \{L, 0, R\}^2$
- s_+, s_- are accept., reject. halting states

Lemma 5:

If a deterministic 2-T-TM M terminates on input x then it holds

$$2\text{-}T\text{-}DTIME_M(x) \leq |x| \cdot 2^{|M| \cdot 2\text{-}T\text{-}DSPACE_M(x)}$$

Proof:

Observe: If M stops then no configuration could be visited twice by M.

Define configuration $=_{def}$ (work. tape inscriptions, pos. of heads, state)

That is:

$$
\begin{aligned}
\text{2-T-DTIME}_M(x) &\le \text{number of configurations} \\
&= (\#\text{ w.t. inscr.}) \cdot (\#\text{ w.t. head}) \cdot (\#\text{ i.t. head}) \cdot (\#\text{states}) \\
&\le_{\|\Delta\| = m} m^{\text{DSPACE}_M(x)} \cdot \underbrace{\text{DSPACE}_M(x)}_{s =_{def}} \cdot |x| \cdot k \\
&\le |x| \cdot 2^{(\log m) s + \log s + \log k} \\
&\le |x| \cdot 2^{s(\log m + \log k + 1)} \\
&= |x| \cdot 2^{s \cdot \log 2mk} \\
&\le |x| \cdot 2^{mks} \le |x| \cdot 2^{|M| s} \quad \blacksquare
\end{aligned}
$$

Corollary 8.

$$\text{DSPACE}(s) \subseteq \text{DTIME}\left(2^{O(s)}\right) \text{ for } s(n) \ge \log n.$$

Proof: $|x| \cdot 2^{|M| \cdot s(|x|)} \le 2^{|M| \cdot s(|x|) + \log |x|} \le 2^{(|M|+1) \cdot s(|x|)}$ $\blacksquare$

Theorem 7

For every computable function $s: \mathbb{N} \to \mathbb{N}$, there is a decidable language L s.t. $L \notin DSPACE(s)$

Proof: Let $M_0, M_1, M_2, \ldots$ be an enumeration of all 2-T-TM; Mapping $i \mapsto M_i$ is computable. Define

$$L_s =_{def} \{ x \mid M_{|x|} \text{ does not accept } x \text{ in space } s(|x|) \}$$

We show that $\overline{L_s}$ is decidable: Consider any TM that on x

(a) computes program of $M_{|x|}$,

(b) computes $s(|x|)$,

(c) computes $|x| \cdot 2^{|M| \cdot s(|x|)}$,

(d) simulates $M_{|x|}$ on x at most $|x| \cdot 2^{|M| \cdot s(|x|)}$ steps

(e) behaves according to the following cases:

- (i): $M_{|x|}$ does not halt: rejects x
- (ii) $M_{|x|}$ halts but rejects x: rejects x
- (iii) $M_{|x|}$ halts and accepts x:
  - if $M_{|x|}$ halts in running space $s(|x|)$: accepts x
  - if $M_{|x|}$ needs more than $s(|x|)$ space: rejects x

It follows that L_s is decidable.

Assume that $L_s \in 2\text{-}T\text{-}DSPACE(s)$, i.e., there is an M_i accepting L_s in space s. Choose x with $|x| = i$. Then,

$$\begin{aligned} x \in L_s &\Leftrightarrow M_i \text{ accepts } x \text{ in space } s(|x|) \\ &\Leftrightarrow M_{|x|} \text{ accepts } x \text{ in space } s(|x|) \\ &\Leftrightarrow x \notin L_s \end{aligned}$$

↯

A function $s: \mathbb{N} \to \mathbb{N}$ is said to be space-constructible iff there is a 2-T-TM M s.t. $DSPACE_M(x) = s(|x|)$.

Proposition 8.

Let $s: \mathbb{N} \to \mathbb{N}$ be a function. Then, the following equivalence holds:

$$s \text{ is space-constructible} \iff s \circ len \in FDSPACE(s)$$

Remarks:

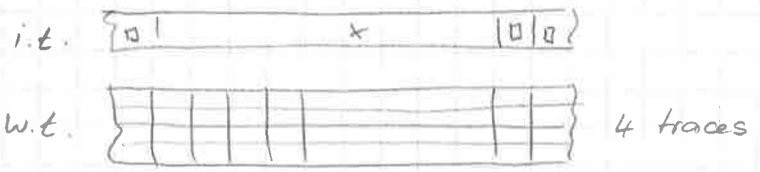
1. $len: \Sigma^* \to \mathbb{N} : x \mapsto |x|$
2. $s \circ len: \Sigma \to \mathbb{N} : x \mapsto s(len(x)) = s(|x|)$

Examples: n^k for $k \geq 1$, $\log n$, 2^n are space-constructible.

Theorem 9. (Hierarchy Theorem)

Let s, s' be space-constructible, $s(n) \geq \log n$. If $s' = o(s)$ then $DSPACE(s') \subset DSPACE(s)$

Proof: Let $M_0, M_1, M_2, \dots$ be an enumeration of all 2-T-TM.
Define M to be the following 2-T-TM, on input x,



Phase 1: simulates a 2-T-TM for constructing space $s(|x|)$; space is marked

Phase 2: if $x \neq 0^m 1^i$ then M halts and rejects x

Phase 3: $(x = 0^m 1^i)$ M constructs program of M_i in trace 2, if not longer than $s(|x|)$, otherwise M halts and rejects x

Phase 4: M simulates M_i in trace 3 in space $s(|x|)$ for at most $2^{2s(|x|)}$ steps of M_i; otherwise M halts and rejects x

Phase 5: M accepts $x \iff M_j$ rejects x

M decides any language L in space $s(|x|)$, i.e., $L \in DSPACE(s)$

Assume $L \in DSPACE(s')$, i.e., there is a 2-T-TM M_j accepting L in space s'

Choose m large enough so that for $n = m+j$ the follow. is true:

(i) $|M_j| \leq s(n)$

(ii) $|M_j| \cdot s'(n) \leq s(n)$ (note: $s' = o(s)$)

Consider M on $x = 0^m 1^j$, $|x| = n$:

- phase 1: M can't stop
- phase 2: M does not stop, since $x = 0^m 1^j$
- phase 3: no stop, since $|M_j| \leq s(|x|)$
- phase 4: M_j runs in space $s'(n) \leq \frac{s(n)}{|M_j|}$ and runs in $|x| \cdot 2^{|M_j| \cdot \frac{s(|x|)}{|M_j|}} = |x| \cdot 2^{s(|x|)} \leq 2^{2s(|x|)}$ steps

That is, M runs in space $s(|x|)$ and simulates M_j. Hence,

M accepts $x \iff M_j$ rejects x

$\lightning$

$\blacksquare$

Theorem 10. (Linear compression)

For every function $s: \mathbb{N} \to \mathbb{N}$,

$$DSPACE(s) = DSPACE(O(s)).$$

Proof: Simulating k cells in one cell leads to compression factor $\frac{1}{k}$. ∎

Remarks:

1. Gap theorem (Trakhtenbrot 64, Borodin 72): For every computable $r: \mathbb{N} \to \mathbb{N}$ there is a computable $s: \mathbb{N} \to \mathbb{N}$ s.t. $DSPACE(s) = DSPACE(r \circ s)$.
2. There is an s s.t. $DSPACE(s) = DSPACE(2^s)$.
3. $REG = DSPACE(1) = DSPACE(s)$ for $s = o(\log \log n)$.

1.2.2 Nondeterministic space

We consider $NSPACE = 2\text{-}T\text{-}NSPACE$.

Theorem 11. (Hierarchy theorem)

Let s, s' be space-constructible, $s(n) \geq \log n$.

If $s'(n+1) = o(s(n))$ then $NSPACE(s') \subset NSPACE(s)$ ∎

Remarks:

1. Diagonalization does not work for nondeterministic measures.
2. Proof is based on recursive padding: For a set A and a function $r(n) > n$ define

$$A_r =_{df} \{ x b a^{r(|x|) - |x| - 1} \mid b \neq a, x \in A \}$$

Then,

$$A_r \in NSPACE(s) \iff A \in NSPACE(s \circ r)$$

(3) Upward translation of equality: We show

$$NSPACE(n^2) \subseteq NSPACE(n) \Rightarrow NSPACE(n^3) \subseteq NSPACE(n)$$

Let $r(n) =_{def} n^{3/2}$, so $n^3 = (n^{3/2})^2 = r^2$. Then,

$s(n) = n^2$, $r = n^{3/2}$

$$\begin{aligned} A \in NSPACE(n^3) &\Rightarrow A_r \in NSPACE(n^2) \\ &\Rightarrow A_r \in NSPACE(n) \\ &\Rightarrow A \in NSPACE(n^{3/2}) \\ &\Rightarrow A \in NSPACE(n^2) \\ &\Rightarrow A \in NSPACE(n) \end{aligned}$$

(4) Savitch's theorem: $NSPACE(s) \subseteq DSPACE(s^2)$ (later!)

We obtain:

$$NSPACE(s) \subseteq DSPACE(s^2) \subset DSPACE(s^3) \subseteq NSPACE(s^3)$$

In particular: $NSPACE(n) \subset NSPACE(n^2)$

Theorem 12. (Linear compression)

For every $s: \mathbb{N} \to \mathbb{N}$,

$$NSPACE(s) = NSPACE(O(s)).$$

■

1.2.3 Deterministic time

Theorem 13. (Hierarchy theorem)

Let t, t' be „easy-to-compute" functions, $t(n) \geq n$.

If $t' = o(t)$ then $\text{multiT-DTIME}(t') \subset \text{multiT-DTIME}(t \cdot \log t)$.

$\blacksquare$

Remarks:

1. Proof is by diagonalization; $\log t$ factor because of simulation of arbitrarily many tapes by a fixed number of tapes.
2. $\text{multiT-DTIME}(t') \subset \text{multiT-DTIME}(t \cdot \sqrt{\log t})$ using recursive padding.
3. $k\text{T-DTIME}(t') \subset k\text{T-DTIME}(t)$ for $k \geq 2$ fixed
4. $\text{RAM-DTIME}(t) \subset \text{RAM-DTIME}(c \cdot t)$ for some $c > 1$.

Theorem 14 (Linear speed-up)

For $t(n) \geq c \cdot n$, $c > 1$,

$$\text{multiT-DTIME}(t) = \text{multiT-DTIME}(O(t)).$$

$\blacksquare$

Remark: $\text{multiT-DTIME}(n) \subset \text{multiT-DTIME}(O(n))$ via

$$L =_{\text{def}} \{ 0^{n_1} \# \ldots \# 0^{n_k} *^m 0^{n_{k-m+1}} \mid k, n_1, \ldots, n_k \geq 1, m \in \{1, \ldots, k\} \}.$$

1.2.4 Nondeterministic time

<u>Theorem 15.</u> (Hierarchy theorem)

Let t, t' be „easy-to-compute" functions, $t(n) \geq n$, $t'(n) \geq n$.
If $t'(n+1) = o(t(n))$ then $\text{multiT-NTIME}(t') \subset \text{multiT-NTIME}(t)$.

■

<u>Theorem 16.</u> (Linear speed-up)

For $t(n) \geq n$,

$$\text{multiT-NTIME}(t) = \text{multiT-NTIME}(O(t))$$

■

<u>Remarks:</u>

1. Linear speed-up already for $t(n) = n$
2. $\text{multiT-DTIME}(n) \subset \text{multiT-NTIME}(n)$
3. $\text{multiT-DTIME}(O(n)) \subset \text{multiT-NTIME}(O(n))$